March 1993

OS-9 International

Issue 1/93

Published by:
Marc Balmer
Hagentalerstrasse 12
CH – 4055 Basel

Phone +41 61 43 55 01
Fax +41 61 43 55 02
E-Mail os9int@msys.ch
ISSN 1019-6714

Editors:
Marc Balmer
Carsten Emde

startup

Finally..., this is the second issue of OS-9 International. As you may have noticed, it appeared somewhat later than initially planned. This is mainly due to the fact that some authors who initially promised to send a manuscript never submitted one. We do hope that future issues will no longer suffer from having underestimated this human factor.

This year will be an important year for OS-9. There are many other competitors than just Microware pushing their products into the real-time market. LynxOS, just to mention one of them, has become the most important manufacturer for the US Department of Defense as far as real-time operating systems are concerned. But competition has always been to the users' benefit: The upgrade changes that Microware recently made (refer to Douglas Kemp's article in this issue), appear to be the first answer to the market changes; more changes must follow, if Microware wants to continue its role in the real-time game.

OS-9 International certainly will continue to support OS-9 by publishing know-how for its efficient use, stay with us!

Carsten Emde

What is EFFO ?

EFFO stands for "European Forum For OS-9" and was founded in 1988; its main goal is to support Microware's multi-user multi-tasking operating system OS-9 that runs on the 68K family of Motorola microprocessors. This support primarily consists in providing a means of communication between people who already use and appreciate OS-9 and those who do not yet but would profit by doing so.

Initially, most of EFFO members were from that rare species of computer owners who were proud of being incompatible with the rest of the world by using the hand-made GDOS on GEPARD computers; the only way to survive was to create a forum for exchanging hardware and software know-how. Later on, OS-9 was ported to the GEPARD computer; this improved the performance of GEPARD computers a lot — but they were still incompatible with the rest of the world. On the other hand, OS-9 was running on a variety of other hardware platforms so they were at least compatible to other incompatibles. In consequence, EFFO was expanded to cover OS-9 in a more general, i.e. hardware-independent, way.

In accordance with the original objective, EFFO intends

- to provide a collection of public domain software that is of general interest and that helps to make OS-9 more attractive to programmers and users,
- to coordinate ports of (mainly UNIX) software (e.g. to avoid that a particular software is ported many times and other equally important software still is not),
- to make all the "nuts and bolts" of managing an operating system available to everybody so that "the wheel has not to be re-invented all the time".

Until the end of 1992, the rules of the game were that disks containing the EFFO software pool were available without charge, and everybody taking part in this service was expected to make contributions to the pool. Initially, this concept – although somewhat idealistic and similar to an economic system that has recently been abandoned – worked quite well. Over the years, however, less and less contributions have been made so that, starting in 1993, a new concept was created.

First the bad news: The distribution is no longer free of charge, but there is a handling fee in an order of magnitude comparable to other PD pools for other operating systems. This issue of OS-9 International contains a complete list of the currently available PD software.

And now the good news:

1. The disks contain ready-to-use software that has been tested thoroughly.
2. The software is continuously maintained and updated.
3. All disks come with printed guidelines of how to install and to use the software. Some of them even have a complete user's manual in printed form.
4. The 23 forum editions and the 10 PD disks representing the EFFO software pool as at end 1992 will – although no longer maintained and upgraded – still be available.

In addition, EFFO now has a printed forum: every issue of the newly founded journal OS-9 International contains several pages with EFFO information including the most recent version of its public domain software list. This software list will also be made available on the EFFO bulletin board and through regular postings to international network boards.

Please contact EFFO at

EFFO
P.O. Box
CH-8606 Greifensee
Switzerland

to obtain more information. This is also the address where all active EFFO members can be reached in case of questions concerning particular software packages.

Last but not least we invite you to join EFFO. As a regular member you get some price reduction on our PD disks, and a one-year subscription to OS-9 International is included.

Protokoll der EFFO GV 1993

Reto Peter

Die Generalversammlung von EFFO (Europäisches Forum Für OS-9) fand am Samstag, 6. März 1993, von 15.00 bis 19.00 Uhr im Restaurant Hegibach in Zürich statt. Mit einem persönlichen Brief inklusive Traktandenliste wurden alle eingeschriebenen Mitglieder von EFFO eingeladen. Anwesend und stimmberechtigt waren folgende Personen (in alphabetischer Reihenfolge) :

Marc Balmer, Hans-Werner Bippus, Beat Forster, Thomas Früh, Guido Hollenstein, Philip Mächler, Hubert Nehring, Gerd Oxé, Heinz Pauli, Reto Peter (Protokoll), Thomas G. Schiele, Werner Stehling (Leitung), Lukas Zeller

Folgende EFFO-Mitglieder waren entschuldigt abwesend :

Carsten Emde (Kurzbesuch als Solidaritätsbezeugung), Stephan Paschedag

Begrüssung

Der Präsident Werner Stehling begrüsst die Anwesenden zur 6. GV von EFFO. Es sind erfreulich viele Teilnehmer präsent.

Wahl der Stimmenzähler

Als Stimmenzähler wird Thomas Schiele vorgeschlagen und gewählt. Es wird festgestellt, dass alle Anwesenden stimmberechtigt sind (12 Teilnehmer).

Genehmigung der Traktandenliste

Die von Werner Stehling vorbereitete Traktandenliste wird einstimmig akzeptiert, es werden keine Zusatzanträge gestellt. Die Reihenfolge wird nach Prioritäten umgestellt. Die Punkte 12 und 13 werden früher behandelt (nach Punkt 9), da der Hauptbetroffene etwas früher gehen muss.

Genehmigung des Protokolls der letzten GV

Das Protokoll der GV 1992 wurde via Forum verteilt und findet keinen Widerspruch. Es wird einstimmig akzeptiert.

Jahresbericht des Vorstandes

Der Präsident Werner Stehling referiert über das vergangene Jahr. Die letzte Ausgabe des Forum war die Nummer 23. Das ganze Jahr über war die Beteiligung am Forum konstant schlecht und die eingesandten Disketten oft auch leer. Darum wurde die Zusammenstellung und Verteilung von Forumsdisketten eingestellt.

Es wurde nach Alternativen Ausschau gehalten, wie OS-9 gefördert werden kann und wo ein Bedarf nach OS-9 SW besteht. Das Zielpublikum im Privatanwenderbereich wurde nach unserer Einschätzung und Erfahrung immer kleiner, vor allem auch, da die für den Privatgebrauch verwendbare "OS-9 HW" (Atari, c't68k, mc68000, Gepard etc.) stark zurückging. Damit ergab sich beim Zielpublikum automatisch eine Verschiebung in Richtung Hochschule und professionelle Anwender.

Die Zykluszeit der Forumsdisketten war relativ gross, der Trend geht darum heute mehr in Richtung Electronic Mail und File Transfer. OS-9 SW wird weiterhin angeboten, neu aber nur noch in Form von PD-Disketten. Als Quelle wird vor allem Unix-SW benutzt, die meist ohne grosse Änderung bzw. Aufwand auf OS-9 lauffähig ist. Für jede PD-Diskette ist eine ganz bestimmte Person verantwortlich, auch für das Nachführen/Aufdatieren der SW auf der PD-Diskette.

Die alten Forums- und PD-Disketten können nach wie vor bezogen werden, von EFFO oder via FTP-Server.

In der Einladung sind nochmals alle Redaktoren namentlich aufgeführt, die in der Vergangenheit in EFFO mitgearbeitet haben. Ein herzliches Dankeschön an alle für die investierte Zeit und Energie.

Wir hatten zwei Mal Besuch im vergangenen Jahr. Es waren dies Kei Thomson, ein sehr aktiver OS-9 Benutzer und Programmierer (u.a. TEX-Portierung) aus Hamburg und vom CoVe Vorstand, Otto Tutzauer und Helmut Kohl aus Frankfurt. An diesem zweiten Treffen wurde intensiv über OS-9 und seine Zukunft diskutiert; dabei kamen wir zum Schluss, dass OS-9 sicher auch in Zukunft

aktuell bleiben wird. Allerdings ist der Support von Microware durchaus verbesserungsfähig, und auch OS-9000 hat bisher nicht den erwarteten Erfolg gehabt.

Die EFFO-Mitgliederzahl hat im letzten Jahr stagniert, was angesichts der Perestroika nach 5 intensiven Forumsjahren und der deshalb reduzierten Aktivitäten von EFFO nach aussen auch nicht verwunderlich ist. Einige Reaktionen gab es bereits auf Werner Stehlings Einladung zur GV, sowohl positiv wie negativ. Zur Zeit hat EFFO ca. 36 Mitglieder, davon 3 Firmen.

Bericht von Kassier und Revisor

Philip Mächler kommt erst später, er kann somit die Jahresrechnung 1992, die mit der Einladung verschickt wurde, nicht präsentieren. Der Präsident spricht dem Kassier den Dank für die sehr gut geleistete Arbeit aus.

Der Revisor Stephan Paschedag ist abwesend, er hat aber einen schriftlichen Bericht an den Präsidenten abgegeben. Nach Prüfung der Rechnung und der Belege beantragt der Revisor die Jahresrechnung der GV zur Annahme. Diesem Antrag wird einstimmig entsprochen.

Wahlen

Zu den bereits in der Traktandenliste notierten Kandidaten fanden sich keine zusätzlichen mehr. Alle vorgeschlagenen Kandidaten wurden klar und einstimmig (wieder)gewählt. Es sind dies :

Präsident	Werner Stehling	(bisher)
Vizepräsident	Reto Peter	(bisher)
Aktuar/Protokollführer	Reto Peter	(bisher)
Kassier	Lukas Zeller	(neu)
Revisor	Stephan Paschedag	(bisher)

Damit sind alle Chargen von EFFO besetzt. Im Namen aller Mitglieder möchte ich allen Kandidaten für Ihre Bereitschaft, ein Amt zu übernehmen, danken.

Was passiert mit OS9 International?

Im vergangenen Jahr ist leider nur eine Ausgabe von OS-9 International herausgekommen. Die Gründe dafür sind einerseits die grossen Verzögerungen beim Eingang der Artikel für die zweite Nummer, andererseits die finanziellen Probleme durch den sehr schwachen Eingang von Werbeaufträgen. Marc Balmer erhofft sich von der zweiten Ausgabe eine Signalwirkung auf die Industrie, und damit einen positiven Trend für den zukünftigen Inserateeingang.

Falls der EFFO-Beitrag an OS-9 International genehmigt wird, kann die Ausgabe 2 sofort erscheinen. Werner stellt zur Diskussion, ob die Zeitschrift nicht billiger produziert werden kann (z.B. kopieren anstatt drucken). Durch die graphischen Darstellungen in der neuen Ausgabe ist Photokopieren allerdings nicht ratsam. Die Produktionskosten der zweiten Ausgabe sind mit SFr. 2000.00 budgetiert, ein Drittel ist zur Zeit durch Inserate gedeckt. Die erste Ausgabe wurde auch an verschiedene Firmen verteilt, damit sie bei HW-/SW-Lieferungen an Kunden beigelegt werden kann. Es wird verabredet, dass bis zum 1. April die Ausgabe 2 herausgegeben wird (mit dem finanziellen Beitrag von EFFO).

Statutenänderung

Auf grund der Änderungen in EFFO wird eine Anpassung der Statuten notwendig. Folgende Neuformulierung wird vorgeschlagen:

Alte Fassung

Art. 5 Das EFFO gibt in regelmässigen Abständen Datenträger heraus. Auf diesen werden auch die offiziellen Mitteilungen publiziert.

Neue Fassung

Art. 5 Offizielle Mitteilungen von EFFO werden in OS-9 International publiziert.

Die Änderung wird einstimmig genehmigt.

Neue Aufgaben für EFFO

Dieses Thema wurde schon unter anderen Punkten angesprochen (siehe weiter vorne in diesem Protokoll).

Das EFFO-Image ist immer noch verbesserungsfähig. Wichtig ist in erster Linie die prompte Erledigung von Bestellungen. Weiter sollte wieder etwas in Richtung Werbung unternommen werden, Beiträge in Zeitschriften wie iX oder anderen Unix-orientierten Zeitschriften. Die Präsenz an Ausstellungen ist ebenfalls eine Möglichkeit, hängt jedoch von der Verfügbarkeit von Mentoren, Vertretern von EFFO und dem Ausstellungsort ab. In jedem Fall soll nicht mehr angekündigt bzw. veröffentlicht werden als nachher auch gehalten werden kann.

Wichtig ist eine ständige Präsenz als Informationssquelle, z.B. durch regelmässige Statusberichte über EFFO auf elektronischen Netzen. Die Nummer des Postchequekontos auf der Einladung ist unvollständig, korrekt ist die Nummer **80-48254-4**.

Das Thema "OS-9 Workshop in Basel" wird an der nächsten Redaktionssitzung diskutiert.

Budget, Jahresbeitrag und PD Gebühren

Kommentierung des Budget durch Werner Stehling. Folgende Änderungen am Budget werden gegenüber dem Vorschlag, der mit der Einladung verteilt wurde, vorgenommen:

- Beitrag an Stephan Paschedag für EFFO-Vertretung am CoCo-Fest in Chicago: SFr. 300.00.
- Reduktion "unvorhergesehene Ausgaben" auf SFr. 200.00.
- Das Vermögen beträgt Ende '92 SFr. 1887.40 und fliesst ins Budget '93 ein.
- Der an der letzten GV beschlossene, jedoch nie angeforderte Beitrag von SFr. 300.00 an OS-9 International kommt noch dazu.
- Der ZERR (c't68k) muss noch bezahlt werden (SFr. 800.00).

Alle Forumsausgaben und früheren PDDisketten sind zu den alten Bedingungen (SFr. 20.00 resp. SFr. 10.00 für EFFO-Mitglieder pro Ausgabe) sowie über ftp erhältlich.

Alle neuen PDs von EFFO sind nicht via FTP erhältlich. Folgende Preise wurden festgelegt :

- SFr/DM 12.00 pro Diskette; SFr/DM 8.00 für EFFO-Mitglieder
- SFr/DM 10.00 Versandkostenpauschale
- Sonderpreise für grosse Pakete
- alternative Medien für die Softwareverteilung (DAT, SyQuest)

Mitgliederbeitrag für 1993:

- SFr/DM 80.00 für Einzelmitglieder
- SFr/DM 150.00 für Kollektivmitglieder

Die Mitgliederbeiträge werden einstimmig genehmigt. Das Budget wird einstimmig genehmigt.

Für die deutschen EFFO-Mitglieder bzw. Besteller von PD-SW besteht ein neues Konto in Deutschland. Damit wird das Einzahlen aus Deutschland vereinfacht.

Internationale Kontakte

In den letzten paar Wochen ergaben sich enge Kontakte zu Peter Tutelaers, der einen niederländischen OS-9 Club (EUROS9) vertritt. Als offizieller EFFO-Vetreter hat sich Stephan Paschedag angeboten am CoCo-Fest in Chicago teilzunehmen. Seine Reise soll zum grössten Teil durch Spenden via Peter Tutelaers finanziert werden. Dieses Fest findet am 1. Mai statt.

Es wird festgehalten, dass EFFO sich primär auf Europa konzentrieren soll (wir sind ja schliesslich auch ein europäischer Verein). Die Verbreitung von EFFO-PDs auf anderen Kontinenten wird geeigneten Usergruppen überlassen, das Ziel unsererseits sind Vereinbarungen mit den entsprechenden Gruppen in USA, Australien und Japan, die auf Gegenseitigkeit beruhen, d.h. wir können auch die entsprechenden PDs von anderen Gruppen übernehmen und hier in Europa verteilen.

Aufbau eines OS-9 Netzes, ZERR

Es existiert bereits ein OS-9 Netz hier in der Schweiz zwischen Basel und Zürich (Marc Balmer und Carsten Emde).

Die beiden oben erwähnten Rechner sind A-Hosts gegenüber dem Internet (24h online, direkter Internet Zugang, mind. 9600 Baud-Modem). Weiter gibt es B-Hosts (24h online, Verbindung zu einem A-Host via Modem) und C-Hosts (alles übrige). Gemäss den Verhandlungen mit den Internet-Betreibern bekommen wir verbindungszeitabhängige Gebühren anstelle von Gebühren, die von der Datenmenge abhängig sind.

Für einen Host ist mit monatlichen Kosten von ca. SFr. 50.00 zu rechnen; die genauen Kosten werden von Marc Balmer bis zur nächsten Redaktionssitzung geklärt. Die notwendige SW für alle Hosts wird zur Verfügung gestellt. Der EFFO-Redaktionsrechner (ZERR) soll auch als Knoten in dieses Netz eingebunden werden (B-Host).

Es wird einstimmig begrüsst, für EFFO eine Internet Adresse (user@effo.ch) zu beantragen. Für ein einheitliches Auftreten hat Werner Stehling unser EFFO-Logo als ASCII-Datei aufbereitet. Marc Balmer plädiert für maximal vier Zeilen lange Signaturefiles, die den Konferenzbeiträgen angehängt werden. Unbestritten ist das Ziel, als Usergruppe einheitlich aufzutreten. Für ein neues E-mail Logo unter Verwendung von ASCII-Zeichen wird ein Wettbewerb ausgeschrieben, Einsendeschluss ist die nächste Redaktionssitzung. Als Gewinn winkt eine Gratismitgliedschaft in EFFO für ein Jahr.

Gewinnen von neuen Mitgliedern

Vorschlag: Wer drei neue, zahlende Mitglieder wirbt, dem wird der Mitgliederbeitrag für ein Jahr erlassen. Dieser Vorschlag findet allgemeine Zustimmung und das Anmeldeformular wird entsprechend angepasst.

Die EFFO-Handzettel, die von Firmen zu Kundensendungen beigelegt werden können, sollen klar erkennen lassen, dass es sich bei EFFO um eine eigenständige Organisation handelt, die unabhängig von der Versenderfirma ist. Ausserdem sollte auch gleich das Anmeldeformular enthalten sein. Gerd schlägt vor, dies als Postkarte zu realisieren, die beigelegt oder angeklebt werden kann. Er entwirft die Postkarte.

Diverses

Die nächste Redaktionssitzung findet am 6. April um 18.30 Uhr im Institut für Astronomie (Büro WS) statt.

EFFO Budget für 1993

	Einnahmen	Ausgaben
Beitrag an OS-9 International		1000.00
Beitrag an OS-9 International für 1992		300.00
Beitrag für InterEUNet		500.00
500 Disketten		500.00
30 Ringordner für Dokumentationen zu 20.00		600.00
Offene Rechnung für ZERR (ct68k)		800.00
Zentraler EFFO Redaktions-Rechner, HW		1000.00
Werbung		500.00
GV 1994		250.00
Bewirtung von Gästen		150.00
Porto, Mitgliederpost		250.00
Beitrag Besuch CoCo-Fest in Chicago		300.00
unvorhergesehene Ausgaben		200.00
Vorschlag '93		1887.40
Vorschlag '92	1887.40	
Jahresbeitrag von 30 Einzelmitgliedern à 80.00	2400.00	
Jahresbeitrag von 10 Kollektivmitgliedern à 150.00	1500.00	
Erlös aus verkauften PD-Disketten	2000.00	
Erlös aus verkauften Dokumentenordnern	450.00	
Summe	8237.40	8237.40

Peter Dibble at CERN

Douglas Kemp

On October 16th, Peter Dibble from Microware Systems Corp. gave a talk at CERN about the future of OS-9. Here is a summary of his contribution:

Small Kernel and OS-9 3.0

There is a new small (<20kB) kernel in preparation. It is a subset of the full OS-9 kernel with IOMAN, debug support and security checks removed. It is aimed at very high volume embedded applications needing a very cheap kernel. The results of this work will be incorporated in OS-9 3.0:

- Speed enhancements (typically 30%).
- Deterministic response (better control of limits) for the calls relevant to real-time work.
- System state time slicing (uses kernel critical regions). Note that RBF is NOT time-sliced yet.
- Additional service calls (lightweight semaphore and thread mechanisms). This, along with the system state time slicing means that more can be done in interrupt routines.
- Very fast signal intercept routines.
- Minimal saving of registers. Device drivers will need to save the relevant registers to benefit from this. For the time being an optional compatibility bit will save and restore the registers.
- The interrupt vector intermediate jump-table has been removed giving better interrupt response.
- There will be a kernel for each CPU type (68000,68020 etc.) which permits optimum coding and also means there is no need to test for kernel type in the code itself.

- An optional buddy allocator for memory will be introduced. This gives basically the same performance in the average case but is considerable better as one tends towards the worst case.

There is a down-side in that the system data structures have been modified. System calls returning such structures will continue to do so by providing shuffling (at the cost of speed). However a recompilation will use a new call with a direct copy.

Expected time-scales

Small Kernel	Q1 '93
OS-9 3.0	Q2 '93

OS-9000

The new 1.3 release includes the Virtual PC (VPC) environment with Windows 3.1, Extended Memory and DDE. Also there is support for X-Windows/Motif, NFS/RPC, Polytools and a simple, but powerful, Menu Manager.

The next release (1.4) should see a "big win" due to the much better code produced by the new Ultra C compiler.

The question of which RISC computers are targeted next is open.

Networks

The latest release of the Internet support package (1.4) has recently been announced. Work is in hand on backplane TCP/IP drivers and performance enhancements.

There is work going on in ISDN interfacing to OS-9, essentially for multimedia applications. There is some concern in the industry that ISDN may be "too late" as recent developments (eg. MPEG video coding) have shown that reasonable full-motion video could be squeezed through most existing US telephone lines without recabling.

Cross-Development (UniHost and PCBridge)

The present cross-development compiler suite (Unibridge) and line oriented C Source-Level Debugger (Unibug/Srcdbg) will be replaced with a "sexier" product known as UniHost using a graphical interface (X-Windows/Motif) to provide distributed debugging of multiple tasks written in Ultra C.

The debugger functions by having the bulk of the code on the host with just a small stub in the target(s). Profiling will also be available (built into the debugger at the moment but will become part of the compiler in the future). Expected time-scale: Early '93. Future development plans are the provision of a System-State C Source-Level debugger, support for emulators, use of SLIP and backplane TCP/IP plus enhancements to the compiler(s) such as profiling.

PCBridge will have similar enhancements to use Ultra-C, Windows and ethernet (rather than just serial lines).

Multimedia

Microware have a special interest in this field as OS-9 is the kernel integrated into the new CD-I players. Full screen full-motion video should soon be available to complement the existing facilities.

Ultra C

This new ANSI C compiler has a front-end which generates an intermediate code for subsequent linking and code generation. It includes a raft of state-of-the-art optimisation techniques which can span the whole program including the libraries because the optimisations are done over the merged intermediate codes. The intermediate code also allows both the compiler to be retargeted to a new CPU family by modifying the back-end code generator and a compiler for a different language (C++?!) to be produced by changing the front-end.

The code produced executes up to 67% faster than that from the existing compiler (where the biggest gain comes from the floating-point coprocessor code) and, typically, 20% better than that from gcc. The compiler has been heavily tested in the default and debug (-g) modes using the Plum-Hall test suite but

less heavily tested in more esoteric modes (eg. optimise for execution speed with a total neglect of space requirements).

To compile you should have at least 2MB of memory. Cross-compilers already exist for HP, SUN 3 and SUN 4 with other work-station architectures being worked on.

The compiler provides three user interfaces :

- Backwards compatible with the existing compiler. This allows existing code and makefiles to be used without change except for embedded assembly language which has a different syntax. A utility to automate the conversion of embedded assembly language statements is available. However one would be advised to modify the sources to use the new libraries and other facilities for optimum performance.
- An ANSI interface.
- A full interface which allows the user to provide a vast number of possible parameters to each and every stage of the compilation chain.

The compiler is complemented by :

- A new linker which uses an index to directly access library units and is thus much faster (usually about a factor 5) than the present linker which uses a serial search. This indexed access has the second advantage that the ordering of units in a library is irrelevant.
- A new C Source-Level debugger.
- A large number of new (faster) libraries.

There is work taking place on debug enhancements (eg. profiling), further code generation improvements, RISC and 64-bit architecture compilers etc.

Future Plans

POSIX

The relevant standards will be implemented on OS-9000 as they become approved. The principal difficulty is the detailed semantics of "fork" – but this can be handled in OS-9000.

RISC

Implementations of Ultra C and OS-9000 are planned for RISC and other CPU families.

Multi-Processor

The "Hydra project" is going ahead. It uses the Non-Uniform Memory Architecture (NUMA) model. That is to say that each processor can freely use its own memory but there are special calls to access shared memory. This difference is a deliberate policy to avoid hiding the performance penalty of shared memory systems. However it has the great convenience of being cheap and easy to implement – one doesn't need special multiported memory or extensive cache coherency logic.

Which processor a particular process was running on would be specified by the initial caller rather than attempting load-sharing. However a load-sharing routine hook would be available to the user should he want to implement a specific algorithm. The processor number would then be merely part of the process identifier handle. No process migration is foreseen.

There is no reason for the processors to be of the same type but there would be problems if they used a different byte ordering since there is no wish to do automatic software byte swapping; the individual processors could, of course, do this.

The present implementation is simply a P2 (system extension module) to OS-9000. Ideas abound to implement a minimal kernel for DSP's etc. who could forward any complicated kernel request to a full-function CPU for execution.

Douglas Kemp is OS-9 Coordinator at CERN. He can be reached at: Douglas Kemp, OS-9 Coordinator, CN-Division, CERN, 1211 Geneva 23, Switzerland or by E-Mail <kemp@dxcern.cern.ch>.

The MGR window manager

Carsten Emde, Wolfgang Ocker

The user interface of the OS-9 operating system normally consists of a command shell, the computer's output is sent to a text terminal and input is expected from an ASCII keyboard. This is acceptable for programmers and for the development of system software or maintenance of old-fashion software that only needs a text terminal and a keyboard to communicate with the user. This is, however, not acceptable for users and for the development of state-of-the-art software, since this often requires a graphical user interface (GUI). In addition, even system programming may considerably be facilitated if multiple text windows can be inspected simultaneously and if a system-wide cut-and-paste function is available.

In contrast to other operating systems such as Macintosh' OS or Atari TOS that always came with a GUI or such as UNIX and MS-DOS that were supplemented with dedicated graphic systems later on in their life, OS-9 never had a commonly available and standardized GUI. Some hardware manufacturers, however, developed graphical tool packages or libraries in order to make basic graphic functions available under OS-9, but most of them only work on the graphics hardware of that particular manufacturer. On the other hand, window managers such as the X window system or the MGR have been developed independently from specific graphic processors or other hardware requirements. They are available in source code form, and their implementation should be an easy solution to let OS-9 no longer suffer from the lack of a GUI. Unfortunately, these window managers were primarily developed to run under the UNIX operating system that does not limit in any kind the use (and abuse) of system memory and CPU performance. Window managers that run on a standard OS-9 system equipped, for example, with a 16 MHz-clocked 68020 processor, 4 MByte of RAM and a 120 MByte hard disk, are difficult to find; the X window system definitely does not fulfill this requirement. The MGR window manager, however, is much smaller in size and works with much less CPU power than the X window system. It was, therefore, decided to implement the MGR window manager under OS-9, to supplement it with functions that are specific for OS-9 and to a add a high-level library. In addition, standard tools such as text and image browsers, icon, graphic and resource editors and a desktop program had to be provided. As of end 1992, this work has been completed successfully: the

MGR window manager for OS-9 in its current version 1.3 provides a GUI that runs on more than 40 different VMEbus graphic boards (monochrome, 4, 16 and 256 colors) and has a powerful high-level library called MGR/ALib. The latter was used to develop the above named programs that are now part of the MGR distribution.

It was the aim of the present article to demonstrate basic principles of the MGR implementation and of the standard and the high-level libraries; in addition, user programs and tools for MGR programmers will be described.

The MGR implementation

With regard to hardware uniformity, the OS-9 operating system is much different from other operating systems that have a standard GUI: Despite the relatively high number of installed OS-9 systems, only very few of them have an identical hardware configuration and there is no standard recommendation of how to connect graphic subsystems to the operating system. It was, therefore, mandatory i) to completely separate the window manager's hardware-dependent part from its hardware-independent part and ii) to implement a shell interface that allows any already existing OS-9 program to run in a window, to set cursor position and screen attributes and to evaluate keyboard arrow keys. The first was done by introducing the MSD (MGR Screen Driver) concept; the second was already a part of the basic MGR implementation; it has been expanded in order to allow for the independent definition of keyboard arrow keys.

The MSD (MGR Screen Driver) concept

An MSD screen driver is a standard SCF driver written in the language C. A variety of different MSDs exists for different graphic processors such as Hitachi's ACRTC, Intel's i82786, for the QPDM and for graphic boards that have no dedicated graphic processor but use direct access to the video RAM. These drivers are available in source code form including a ready-to-use make environment as part of the MGR/PortPak. Experience from previosly made implementations has shown that most graphic boards with direct access to video RAM could be implemented in less than an hour and most graphic boards with one of the above named already supported graphic processors could be put to work in less than a day. In many cases, is was not necessary to modify the driver but it was sufficient to select appropriate initialization constants. The latter are passed

to the MSD in form of a standard option string, so that this does not involve recompiling.

An MGR screen driver has all the standard Init, Read, Write, GetStt, SetStt, Term entries implemented with the exception that Read and Write do not provide intrinsic functionality but simply return with the uninstalled service request error. Any communication between the window manager and the graphics hardware is done using GetStt and SetStt calls; they represent the main part of an MSD driver. The first steps of the MSD initialization function for the ACRTC controller has, for example, been realized as follows:

```
int _setstat(func, pathdesc)
  int            func;
  union pathdesc *pathdesc;
{
  switch (func) {
    case SS_MSD:
      switch (reg->d[2]) {
        case SS_MSD_INIT:
          init_acrtc(&setup);        /* initialize ACRTC */
          init_video(&video);        /* initialize video structure */
          if (read_options(&setup, &video) < 0) { /* parse options */
              errno = E_BADPARM;     /* error, bad parameters */
              return(-1);
            }
        ...
      }
    ...
  }
}
```

As advantages of using the language C, changes are easy to do even from programmers that are not familiar with the Motorola 68K assembly language, documentation is much facilitated and, nevertheless, this is, realistically, the only way to implement the MGR on other operating systems that run on other processors.

The Pseudo-TTY terminal emulation

The communication between an OS-9 program and an MGR terminal window is done via pairs of pseudo-TTYs so that the program will not necessarily notice that the terminal is not made by hardware but made by software. All programs that are part of the Professional OS-9 package and all other text-oriented

programs will, therefore, run without any restriction in an MGR window. Programs that use the termcap library for positioning the cursor and for setting screen attributes will in most cases also run, since all normally required termcap definitions are available:

al	=	\Ea	Add (insert) blank line
am			Automatic margins
bs			Ctrl-H does backspace
cd	=	\EC	Clear to end of display
ce	=	\Ec	Clear to end of line
cl	=	^ L	Clear screen
cm	=	\E%r%d,%dM	Cursor motion
dc	=	\EE	Delete character
dl	=	\Ed	Delete line
do	=	\Ef	Down one line
ic	=	\EA	Insert character
im	=		Insert mode (enter)
nd	=	\Er	Nondestructive space (cursor right)
se	=	\En	End stand out mode (normal video)
so	=	\Ei	Begin stand out mode (inverse video)
ta	=	^ I	Tab
up	=	\Eu	Upline (cursor up)

In addition, the termcap entries co# and li# are supported. The MGR will set these entries to the current values, whenever a new shell window is created. Such dynamic adaptation is not possible using the static termcap file (normally available as /dd/SYS/termcap), but the TERMCAP environment variable is provided for this purpose, and the MGR makes use of this feature. In addition, the MGR utility set_termcap produces a string that contains the current termcap setting preceded by "setenv TERMCAP":

```
$ set_termcap
setenv TERMCAP "px|mgr|MGR Terminal Emulator:
am:li#24:co#80:bs:cl=^L:ce=\Ec:cd=\EC:cm=\E%r%d,%dM:
al=\Ea:dl=\Ed:ic=\EA:dc=\EE:im=:ta=^I:up=\Eu:do=\Ef:
nd=\Er:so=\Ei:se=\En:"
```

Note that the keyboard arrow key codes are normally not part of this setting; for this purpose, the command "add-termcap=<keyboard>" in the MGR installation file has to be used; the MGR will, if set, add the string "<keyboard>" to the string in the environment variable TERMCAP.

If shell programs are used that support the redefinition of environment variables such as the Bourne shell, the command

```
eval 'set_termcap'
```

will adapt the termcap setting after the window has been resized.

Such programs that do not use the termcap library but have hard-wired terminal control codes as, for example, the VT-100 setting will not work properly within an MGR window. Although a VT-100 terminal emulation program exists, it is recommended to replace the hard-wired codes by appropriate calls to the termcap library, since this is good programming practice anyway.

Low-level programming

For the realization of graphic commands, the above concept for positioning the cursor and for setting screen attributes was expanded but, principally, it still consists of control (mostly escape) sequences that inform the MGR server about the desired window action. Even installing menus and downloading bitmaps work the same way. The only difference to the termcap concept is that the required control sequences are not contained in an external settings file or environment variable but are provided for the programmer in form of macros or functions. In addition, the MGR server may send messages to the program if information such as the current window size is needed or, more importantly, events are to be evaluated in order to inform the program about specific user actions. These two directions of message exchange between a program and the MGR window manager are exemplified in the following program section, where the server is first asked to deliver information about the current screen geometry and the program then generates the command to draw a centered circle on the screen with a radius of one third of the horizontal screen size in landscape or one third of the vertical screen size in portrait format, respectively:

```
main()
{
  int     xmax, ymax;

  ckmgrterm(NULL);     /* check whether really running in an
                          MGR window, abort otherwise */
  m_setup(0);          /* setup MGR communication environment */

  get_size(NULL, NULL, &xmax, &ymax);
```

```
                    /* get screen dimensions, this function reads a
                       message string from the MGR, scans the string
                       and sets the variables xmax and ymax accord-
                       ingly */

  m_setmode(M_ABS);   /* set to absolute screen coordinates */
  m_clear();          /* clear the screen */
  m_circle(xmax / 2, ymax / 2, xmax < ymax ? xmax / 3 : ymax / 3);
                      /* draw the circle */
}
```

High-level programming

The high-level library MGR/ALib provides a set of functions that are needed to create an appropriate GUI. In addition to basic functions such as installation and termination of the MGR/ALib window mechanism, bitmap and menu administration and highly integrated functions such as creating a fully equipped file selector box and automatically sized alert boxes are available. Furthermore, a variety of dialog elements can be selected and positioned anywhere within a dialog box. For this purpose, an MGR resource language, a resource compiler and a resource editor have been created. The requirements of a minimal MGR/ALib application are demonstrated in the following example program that displays the MGR file selector box on screen, lets the user make a selection (or abort) and writes the complete name of the selected file, if any, to standard output path:

```
$ mfsel -?
Syntax: mfsel [<opts>]
Function: MGR File selector program
          mfsel will display a file selector box and write
          the file selected, if any, to standard output path
Options:
     -d=<dir> initial directory will be <dir>
     -e=<ext> the extension <ext> will be displayed
     -h=<txt> the text <txt> will be shown in the header of the
              file selector box
     -n       the NONEW flag will be set, only existing files can
              be selected

/*
 * m a i n
```

```
 */
main(argc, argv)
  int  argc;
  char *argv[];
{
  char         *dir = NULL, *ext = NULL, *file = NULL, *header = NULL;
  char         defdir[512];
  register int i, j;
  int          window, nonew = 0;

  /*
   * parse arguments
   */
  for (i = 1; i < argc; i++)
    if (argv[i][0] == '-')
      for (j = 1; j < strlen(argv[i]); j++)
        switch (argv[i][j]) {
          case '?':
            usage();
            exit(1);
          case 'd':
            dir = argv[i] + j + (argv[i][j+1] == '=' ? 2 : 1);
            j   = strlen(argv[i]);
            break;
          case 'e':
            ext = argv[i] + j + (argv[i][j+1] == '=' ? 2 : 1);
            j   = strlen(argv[i]);
            break;
          case 'h':
            header = argv[i] + j + (argv[i][j+1] == '=' ? 2 : 1);
            j      = strlen(argv[i]);
            break;
          case 'n':
            nonew = FS_NONEW;
            break;
          default:
            usage();
            exit(_errmsg(1, "unknown option '%c'\n", argv[i][j]));
        }
    else {                          /* parameters */
      usage();
      exit(_errmsg(1, "no parameters recognized\n"));
    }

  /*
   * Setup everything
   */
  ml_setup();
```

```
  if ((window = ml_create_window(0, -1, -1, -1, -1, NULL, 0)) < 0) {
    ml_cleanup();
    exit(_errmsg(1, "can't allocate main window\n"));
  }

  /*
   * Now open the file selector box
   */
  file = ml_fsel(dir == NULL ? dir : strcpy(defdir, dir), "*",
                 ext == NULL ? "*.txt" : ext,
                 header == NULL ? "Please select file" : header,
                 nonew, -1, -1, -1);

  ml_cleanup();                        /* cleanup everything */

  /*
   * Write filename to standard output
   */
  if (file != NULL && file[0] != '\0') {
    printf("%s\n", file + 1);
    exit(0);
  }
  exit(1);
}
```

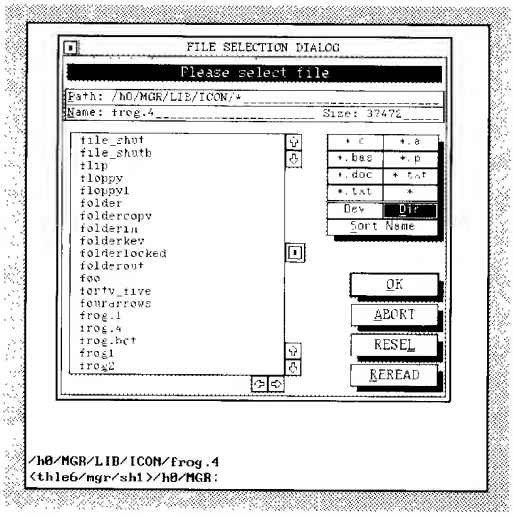

Figure 1: Screen hardcopy of the above example

Figure 1 shows an MGR screen hardcopy that has been taken after running the above example programs on a monochrome graphic board.

Tools for the user

A wide variety of user programs have been developed by using the MGR/ALib; it would exceed the scope of this article to present all of them. The following three programs, a text and an image browser and the MGR desktop, have been selected arbitrarily to serve as examples.

The Mmore MGR text browser

The Mmore text browser accepts input from the standard input channel or from one or more files passed as arguments to the program. Mmore uses a standard MGR/ALib window with a title above the window, and a vertical and a horizontal scroll bar. In addition, buttons below the title bar are provided that offer a number of specific functions:

Mmore – Display about box containing information about the program
Next – Show next text file
Prev – Show previous text file
List – Pop-up a menu with all text files, show the selected one
Find – Find search pattern
FRes – Reset search pattern
FNext – Search forward
FPrev – Search backward
Cut – Cut part of the text and put it to the system's snarf buffer
Goto – Enter line number to become the first row
Pos – Show current line number, last line number, percentage
Tabs – Define tabulator width
Font – Pop-up a menu with available fonts, use the selected one

Figure 2 shows an MGR screen hardcopy that has been taken after starting the Mmore program with Microware's good old message file /dd/SYS/motd as argument, clicking on the "Find" button and searching for all occurences of the string "OS-9". The MGR server was running on a 4-bit graphic board with the color look-up table set to 16 equidistant grey-scale levels.

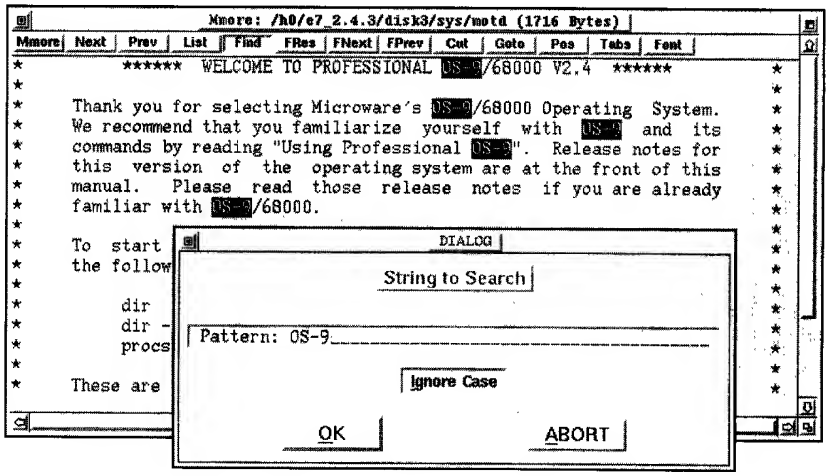

Figure 2: Screen hardcopy of mmore

The Mpan MGR image browser

The MPan image browser is similar to the above text browser; it has the following functions:

Mpan – Display about box containing information about the program
PosI – Show current cursor position relative to the upper left corner
Next – Display next image
Prev – Display previous image
List – Pop-up a menu with all images, display the selected one
Full – Resize the window to full size of the current image
Step – Select the step width for scrolling
Inv – Invert the color numbers

Figure 3 shows an MGR screen hardcopy that has been taken after starting the Mpan program with the MGR bitmap file "golfer.g4". The latter had been created previously using the GhostScript for OS-9 program; the demo script "golfer.ps" is part of that package. As in Figure 2, the MGR server was running on a 4-bit graphic board set to 16 grey-scale levels. In contrast to Figure 2, the Mpan application was compiled without the 3D-flag in order to demonstrate these two different basic styles the MGR/ALib is capable of generating.

Figure 3: Screen hardcopy of mpan

The MGR desktop Mdt

Since there is no obvious reason that Macintosh or MS-Windows users should have better facilities for file manipulation and inspection than OS-9 users, the MGR desktop Mdt was developed, again using the MGR/ALib tools. The following main functions are available:

Mdt	–	Display about box containing information about the program
..	–	Display parent directory
Chd	–	Enter new directory to display
Info	–	Show and optionally edit file attributes
Copy	–	Copy selected files and/or directories to new location
Move	–	Move selected files and/or directories to new location
NFld	–	Create new directory
Del	–	Delete selected files and/or directories
Misc	–	Select all, unselect all, move file string to snarf buffer
View	–	Show selected text and image files using mmore and mpan
Opt	–	Set fonts, security level, file and device icons, aliases etc.
W*.c	–	Define wildcard pattern string for selection
*Icn	–	Toggle icon and text display mode
Clon	–	Duplicate current window

Note that maximum attention was drawn to include as many shell functions as possible such as, for example, wildcard selection and aliasing.

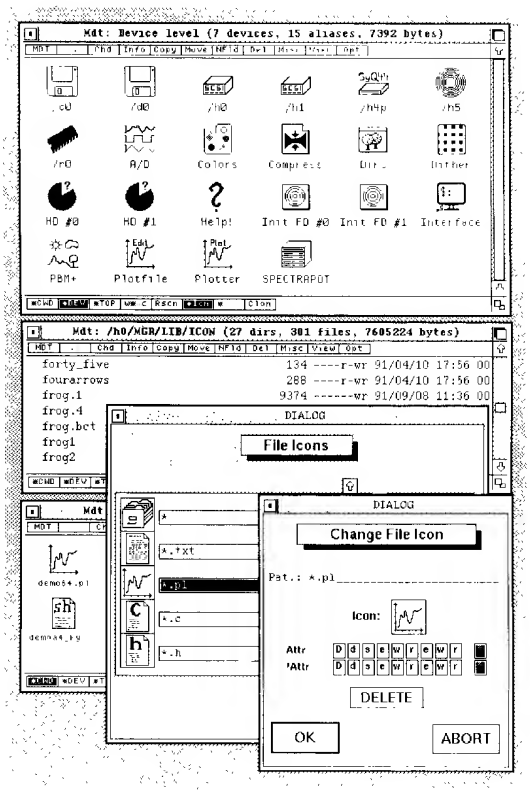

Figure 4: The MGR desktop Mdt

Figure 4 shows an MGR screen hardcopy with three Mdt windows (Device window, above; directory window in text display mode, second from above; directory window in icon display mode, below). In addition, the option function "Change File Icons" was selected and the installation of a file icon is beeing used. The MGR server was running on a monochrome graphic board.

Tools for programmers

Most of the MGR/ALib applications have been developed in less than a week programming time. This would not have been possible if a powerful resource editor had not been available. In fact, the Mre resource editor is the key for the efficiency of MGR/ALib programming.

The Mre MGR resource editor

The Mre MGR resource editor is called with an MGR/ALib resource as argument. In a first step, the object tree is displayed hierarchically; the programmer may then select one of the objects and edit attributes such as type, position, color, strings, fonts etc. More importantly, however, it is possible to have the complete dialog being displayed on screen in the same appearance as later on when the application is called by a user. The programmer may then use the mouse to reposition objects so that even very busy dialog boxes can be edited very quickly. This is greatly facilitated by powerful functions that automatically align objects to each other.

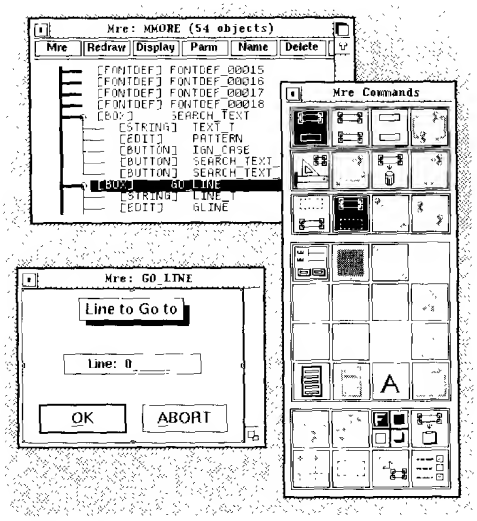

Figure 5: Working with Mre

Figure 5 shows an MGR screen hardcopy that has been taken when editing a dialog box of the Mmore text browser program. The MGR server was running on a monochrome graphic board.

Additional features that are specific to OS-9

In order to make MGR applications as small as possible and to beware the MGR server from "creeping featurism" (the fate of other window managers), two OS-9 specific features have been used: trap handlers and subroutine modules.

Trap handlers

Both the MGR standard library and the MGR/ALib are available in two forms; one that contains all required functions directly within the library and another that uses the mgrtrap and the mgratrap trap handler for standard and for MGR/ALib functions, respectively. Use of these trap handlers leads to a considerable reduction in program memory requirement. Any time, more than about five standard MGR programs or more than about two MGR/ALib applications are running concurrently, less overall memory is required. The following list gives some example program sizes of the two version (values approximated):

Standard MGR program	*Without*	*With trap handler*
Shape (reshape window)	15 kByte	6 kByte
Clock2 (analog clock)	20 kByte	8 kByte
Msnap (screen capture)	25 kByte	16 kByte

MGR/ALib application	*Without*	*With trap handler*
Mmore (Text browser)	112 kByte	35 kByte
Mpan (Image browser)	99 kByte	27 kByte
Mdt (MGR desktop)	199 kByte	113 kByte

Subroutine modules

Should a programmer need a server function that is not currently implemented, he or she may add it to the MGR without actually changing the server itself. This is done by linking one or several OS-9 subroutine modules at run-time and making their functions available to the MGR. These subroutines are called extension modules. The following is an example of adding a function that draws a grid into a given window. The first listing is the functionally important part of the extension module, the second listing shows the call to the newly installed function from a user program.

```
/*
 * g r i d
 *
 * p[0] = dest bitmap
 *
 * p[1] = x, p[2] = y, p[3] = w, p[4] = h
 *
 * p[5] = step x, p[6] = step y
 *
 * p[7] = bitblit mode
 *
 */
void grid(xp, nump, p, str)
  XMOD_PARAM   *xp;
  int          nump;
  register int *p;
  char         *str;
{
  register BITMAP *map;
  register int    i;

  if (p[0] < 0 || p[0] > MAXBITMAPS || p[1] < 0 || p[1] > MAXBITMAPS)
    return;

  map = p[0] == 0 ? xp->win_map : xp->bitmaps[p[0]-1];

  if (map == NULL)
    return;

  if (p[0] == 0) {
    xp->clip = 1;
    xp->x0   = p[1];
    xp->y0   = p[2];
    xp->x1   = p[1] + p[3];
    xp->y1   = p[2] + p[4];
  }

  p[3] += p[1];
  p[4] += p[2];

  for (i = p[1]; i < p[3] ; i += p[5])
    (*xp->bit_linepat)(map, i, p[2], i, p[4]-1, p[7], NULL, 0);

  for (i = p[2]; i < p[4]; i += p[6])
    (*xp->bit_linepat)(map, p[1], i, p[3]-1, i, p[7], NULL, 0);
}

main()
{
```

```
  ...
   if (grid)
    mx_grid(xmod, 0, 0, 0, wide*factor, high*factor,
      factor, factor, B_INVERT|(255<<4));
  ...
}
```

Conclusion

The MGR implementation for OS-9 now represents a uniform, fully equipped and generally available graphical subsystem that has been ported to more graphic boards than any other OS-9 graphic system. The X window system, however, also has been ported to OS-9 and is available in several versions; one of these ports was done by Microware. Which of them, MGR or X window, is recommended?

There are three main criteria for the decision-making between these two window managers:

Criterion 1: Source code compatibility

The total number of already existing MGR programs is certainly much less than the total number of X window programs. If in a given project X window sources have to be used by any reason, then this is a very strong argument in favor of X window (some people believe that this is also a strong argument to directly use a UNIX-derived real-time operating system instead of OS-9).

Criterion 2: "Standard"

If management people want a "standard" and not a somewhat "exotic" graphic system, the attempt should be made to re-evaluate their opinion with facts instead of emotion. If this fails, then this again is an argument in favor of X window.

Criterion 3: Budget (Man power, memory and CPU requirement)

In all cases, when a graphic system is desired that is well tuned to all properties that made OS-9 successful ("smart, powerful and highly efficient development system") and none of the above arguments apply, the MGR should be chosen. When extrapolating the amazing progress MGR for OS-9 made over the last two years, it would not be surprising if the MGR became the standard graphical user interface for OS-9 and the second criterion no longer applied.

Carsten Emde discovered OS-9 at a time when one could only dream of the 68K processor family. He continuously used OS-9 and his today's professional activity covers the development of OS-9 system and user software and system integration of VMEbus based industrial computers. He can be reached by E-mail at <carsten@ce.pr.net.ch>.

Wolfgang Ocker owns the reccoware systems company. He made the OS-9 port to Atari ST computers for Cumana some years ago. Today, his professional activity centers on the MGR window manager that he ported to OS-9. He can be reached at his company in Unterwittelsbach (Germany) (+49 8251 51299).

Writing a device driver in C

Marc Balmer

There are two main reasons why programming OS-9 device drivers is considered difficult and why it may be prone to errors: First, drivers are part of the operating system and run in supervisor mode, so that most of the protection mechanisms are disabled. Secondly, device drivers are traditionally written in the assembly language; such code is harder to read and even extensive commenting does not resolve this problem. This disadvantage of the assembly language is probably the most important reason to use a high-level language for the purpose of driver programming. Another reason is, of course, the fact that high-level code may be ported to another processor type.

As drivers become more complex, for example due to more complex transmission protocols such as in mass-storage devices and for networking, the efficiency of an algorithm is more important than the bare speed that eventually can be achieved by using assembly language. In addition, if execution time is a serious concern, speed limiting parts of the driver may still be written in assembly language but, in most cases, a more efficient algorithm in a high-level language will execute more quickly than a high-speed but less efficient algorithm in the assembly language.

Principally, any high-level language can be used for writing device-drivers, but the C language is best suited for this task under OS-9, since all system definitions are available as ready-to-use header files. Unfortunately, it is not possible to write a device driver in the C language without taking several provisions.

When compiling a C program, a series of subroutines are produced that will later on be called using the "`jsr`" or "`bsr`" instruction. Even the "`main()`" routine finally is nothing more than such a subroutine. In order to produce an executable program, the linker adds a small piece of code – the so-called cstart module – to the output file. This cstart module sets the environment, processes the command line and calls the C program's "`main()`" routine. Since all this is not required in a device driver, the standard cstart module cannot be used for developing device drivers. On the other hand, all OS-9 drivers need a jump table at the execution entry; the fact that such a table is not produced by the standard cstart module is another reason why this cstart is not suitable for device drivers.

When the above-mentioned jump table is used to call a particular device driver entry, registers are differently used as if a C subroutine was called: The OS-9 system code passes parameters in registers, whereas parameters of C functions are passed not only in registers but also on the stack; C subroutines return a result in the D0 register, but system code uses the D1 register for a return value.

It has, furthermore, to be considered that the static memory area in a device driver is always addressed relative to the address register A2: If a variable named "`state`" is defined using

```
 vsect
state: ds.b 1
 ends
```

it must be addressed as "`state(a2)`", which is completely different from the corresponding functionality in C. If, outside a function, "`state`" is declared in C as

```
unsigned char state;
```

the C compiler will translate any reference to it as "`state(a6)`".

Additionally, device drivers must not contain initialized static memory; in consequence, the syntactically correct C statement

```
static unsigned char state = 1;
```

is illegal in a device driver and will lead to an error message during linking.

In conclusion, the first - and most important - prerequisite for writing OS-9 device drivers in the C language is the availability of a cstart module that has been designed especially for this purpose and contains the following functionality:

- A valid OS-9 module header for device drivers is generated.
- A correctly positioned jump table is produced.
- The register contents are adapted to meet the calling conventions for every device driver entry.
- Result values are copied from register D0 to register D1, if applicable. Error conditions in the driver must be flagged by setting the carry bit in the CCR.

A documented assembly language cstart module for SCF device drivers written in the C language is given in the following listing. This code must be modified accordingly, if it is going to be used for device drivers for other file managers such as RBF or NFM.

```
* Assembler Interface for OS-9 SCF device drivers in C
* Copyright (C) 1992,93 by OS-9 International

          nam     cstart_scf
          ttl     cstart for SCF drivers in C

Edition   equ     2
Typ_Lang  set     (Drivr<<8)+Objct
Attr_Rev  set     ((ReEnt+SupStat)<<8)+0

          psect   cstart_scf,Typ_Lang,Attr_Rev,Edition,0,_entry

          use     defsfile

          vsect
Stack     ds.b    2048
StackPtr  equ     *-Stack
          ends

_entry    dc.w    _init Driver jumptable
          dc.w    _read
          dc.w    _write
          dc.w    _getstat
          dc.w    _setstat
          dc.w    _term
          dc.w    0

* INIT Entry

_init     movem.l a4/a6,-(sp)     Prepare arguments
          move.l  a2,d1
          move.l  a1,d0
          move.l  a2,a6           Switch static base
          lea.l   StackPtr(a2),a5 Fake frame pointer
          bsr     Init            Call Init()
          movem.l (sp)+,a4/a6     Restore the stack
          move.w  d0,d1           Get return value
          bne     _error          Yes, error
          rts                     No error

* READ Entry

_read     movem.l a4/a6,-(sp)     Prepare arguments
```

```
          move.l  a2,d1
          move.l  a1,d0
          move.l  a2,a6           Switch static base
          lea.l   StackPtr(a2),a5
          bsr     Read            Call Read()
          movem.l (sp)+,a4/a6
          clr.w   d1              No error
          rts

* WRITE Entry

_write    move.l  d0,-(sp)        The character to write
          movem.l a4/a6,-(sp)     Prepare arguments
          move.l  a2,d1
          move.l  a1,d0
          move.l  a2,a6
          lea.l   StackPtr(a2),a5
          bsr     Write           Call Write()
          movem.l (sp)+,a4/a6
          move.w  d0,d1
          move.l  (sp)+,d0
          tst.w   d1
          bne.s   _error
          rts

* GETSTAT Entry

_getstat  move.l  d0,-(sp)        The GetStat Code
          movem.l a4-a6,-(sp)
          move.l  a2,d1
          move.l  a1,d0
          move.l  a2,a6
          lea.l   StackPtr(a2),a5
          bsr     GetStat         Call GetStat()
          movem.l (sp)+,a4-a6
          move.w  d0,d1
          move.l  (sp)+,d0
          tst.w   d1
          bne.s   _error
          rts

* SETSTAT Entry

_setstat  move.l  d0,-(sp)        The SetStat Code
          movem.l a4-a6,-(sp)
          move.l  a2,d1
          move.l  a1,d0
          move.l  a2,a6
          lea.l   StackPtr(a2),a5
```

```
          bsr     SetStat         Call SetStat()
          movem.l (sp)+,a4-a6
          move.w  d0,d1
          move.l  (sp)+,d0
          tst.w   d1
          bne.s   _error
          rts

* TERMINATE Entry

_term     movem.l a4/a6,-(sp)     Prepare arguments
          move.l  a2,d1
          move.l  a1,d0
          move.l  a2,a6
          lea.l   StackPtr(a2),a5
          bsr     Term            Call Term()
          movem.l (sp)+,a4/a6
          move.w  d0,d1
          bne.s   _error
          rts

* Error return

_error    ori    #Carry,ccr       Return with carry set
          rts

          ends
          end
```

The following syntax applies, if the above cstart code is used:

```
int Init(mod_dev *dd, Scfstatic Stat,
  void *ProcDesc, void *SysGlobs);

int Read(Pathdesc pathd, Scfstatic Stat,
  void *ProcDesc, void *SysGlobs);

int Write(Pathdesc pathd, Scfstatic Stat,
  void *ProcDesc, void *SysGlobs, unsigned char c);

int GetStat(Pathdesc pathd, Scfstatic Stat, void *ProcDesc,
  REGISTERS *Regs, void *SysGlobs, unsigned char Code);

int SetStat(Pathdesc pathd, Scfstatic Stat, void *ProcDesc,
  REGISTERS *Regs, void *SysGlobs, unsigned char Code);

int Term(mod_dev *dd, Scfstatic Stat);
```

The device driver may now be translated using a C compiler in a similar way

any other C program would be treated; it must, however, be ensured that the C compiler front-end prevents the linker from linking the standard libraries to the final output module. This is due to the fact that the standard front end would add the original cstart module instead of the driver cstart module to the output file.

If, for example, the GNU C compiler is used to compile the SCF device driver drivr.c and the cstart module is called cstart_scf.l, the following command has to be entered:

```
$ gcc -nostdlib drivr.c -o drivr -lcstart_scf.l
```

The above sections mainly deal with technical aspects of the C language interface but, to successfully write device-drivers in C, additional restrictions with respect to programming technique and style apply.

Using C libraries

Standard C libraries may only partially be used for device drivers. The main reason for this restriction is that many functions use initialized static variables that are not allowed in drivers. This, obviously, not only applies to the functions themselves but also to functions that are based on them. For example, the character classification and conversion function "`islower()`" uses an initialized static table and any function based on it, such as "`atoi()`", may not be used either.

All C library functions that are unusable because of this reason must be reprogrammed avoiding initialized static memory. There are sources for excellent C code to produce an appropriate surrogate for the clib.l library that does not use initialized static memory: P. J. Plauger's book "The Standard C Library" offers a complete C library in source code form at a more than reasonable price. The functions presented in this book are all in ANSI C and, thus, may not work with a K&R C compiler; they have, however, successfully been used with the GNU C compiler.

Another source for suitable C library functions is the freely available source code of the 4.3 BSD Unix for 80386-based computers; this code is distributed under the GNU general public licence and may, therefore, only be used for private purposes and not for commercial software.

If a function relies on a table, e.g. for speed reasons, and the content of the table is read-only, the following trick may be used: Make the data program-counter relative by including them into a psect section:

```
 psect table,0,0,0,0,0

_table: dc.b "This a read-only string",0

 ends
 end
```

Then add the following declarations to your C driver:

```
extern void _table(void);
unsigned char *table

unsigned char get_table_entry(int num)
{
  if (!table) table = (unsigned char *) _table;
  return table[num];
}
```

If functions from a standard C library are used, it is often necessary to declare the global variable "`int errno`". Many library functions use this variable to indicate which error, if any, occurred. The declaration of the errno variable can be done anywhere in the device driver except within a function.

Static vs. dynamic variables

It is generally considered good programming style to preferably use dynamic variables and to minimize the number of global variables. In addition, it is always recommended to combine global variables, if unavoidable, into structures. These programming paradigms, however, do not apply for device drivers, since a driver's stack memory is limited and combining variables into structures would result in an addressing overhead that may decrease execution speed. In consequence, even a state-of-the-art C language driver may contain a lot of global variables. To declare a global (static) variable, this can either be done by declaring it explicitely as "`static`" or by declaring it outside of functions.

Addressing I/O ports

To address an I/O port using the C language syntax, care must be taken to use the correct access mode, i.e. byte, word, or long access. This is commonly achieved by declaring a pointer or array variable that fits to the required addressing mode. If, for example, an I/O controller's status register (REG) is located at address $FE500000 and can only be read by using word access, the following program section could be used:

```
#define REG 0xFE500000L

unsigned short status()
{
  static unsigned short *reg;

  if (!reg) reg = (unsigned short *) REG;
  return *reg;
}
```

If an ANSI C compiler is used such as the GNU C compiler, it is recommended to declare all variables that reference an I/O address as "`volatile`":

```
#define REG 0xFE500000L

unsigned int status()
{
  volatile static unsigned short *reg;
  static unsigned short val1, val2;

  if (!reg) reg = (unsigned short *) REG;
  val1 = *reg;
  val2 = *reg;
  return val1 + val2;
}
```

If "`*reg`" had not been declared "`volatile`", an optimizing compiler would assume that the contents of "`reg`" do not change during program execution and always assign the same value to "`val1`" and "`val2`".

Debugging C language drivers

The dilemma with debugging C language device drivers is that the C language source level debugger does not run in system state and the system state debug-

ger does not know anything about the C language. Nevertheless, the system state debugger can be used to debug C language device driver at the assembly language level. Unfortunately, the system state debugger is not part of the Professional OS-9 package and must be bought separately. Prior to debugging, it is recommeded to run the compiler and linker with options, so that a symbol table and an assembly listing is generated. The assembly listing can serve as a reference to the original source code; this can greatly be facilitated if the C compiler has an option to insert the corresponding C language line as a comment at appropriate locations within the assembly code.

It must be pointed out that this method is not satisfying, especially when debugging large drivers or when library routines are called that are not available in source code form. Therefore, another method, called "invasive" debugging, may be used. This method makes use of a preprocessor definition, e.g. "DEBUG", that is only defined if the driver is going to be debugged. In this case, special code is included that produces debugging messages about the driver's state. Such debugging code might, for example, write into a data module that can be inspected at run-time by another program. Slow drivers may even open a serial device and write the debugging information to a terminal; it must, however, be ensured that all paths that are opened by the driver are explicitely closed before the driver exits since this is not done automatically by the kernel. The reason for this behavior is that all paths opened by a device driver are system paths whereas paths opened from an application program are user paths.

The choice of the right C compiler

One of the advantages of using the C language for writing device drivers, is, as already mentioned, the possibilty to use highly efficient algorithms. The compiler, however, should be able to produce at least comparable efficient code to maintain this advantage. The author has repeatedly and succesfully used the GNU C compiler (version 1.4.2 and 2.3.3) for the development of various device drivers. This compiler is not only fully ANSI compatible but also produces highly optimized code.

This article does not contain a working example of a device driver in the C language, since a future issue of OS-9 International will present a terminal emulator that is written in C and that may easily be ported to any existing graphics hardware. This driver requires the SCF cstart interface described herein.

Acknowledgement

The author wishes to express his sincere gratitude to The Open Software Foundation for making the GNU C compiler available and to Stephan Paschedag who carefully ported this fine compiler to OS-9 and continuously maintains it.

Marc Balmer has been working with OS-9 for the past five years. His special interests are system and real-time software development. He can be reached by e-mail at <balmer@ifi.unibas.ch>.

Where to get free OS-9 Software

Anonymous FTP

Country	*FTP Address*
Germany	reseq.regent.e-technik.tu-muenchen.de
	tupac-amaru.informatik.rwth-aachen.de
	ifi.informatik.uni-stuttgart.de
	athene.uni-paderborn.de
	sun0.urz.uni-heidelberg.de
Finland	nic.funet.fi
	mcsun.eu.net
	uunet.uu.net
Australia	brolga.cc.uq.oz.au
	kirk.bu.oz.au
Japan	srawgw.sra.co.jp
	toklab.ics.osaka-u.ac.jp
	oskgate0.mei.co.jp
France	irisa.irisa.fr
Sweden	sunic.sunet.se
Switzerland	lucy.ifi.unibas.ch
USA	cabrales.cs.wisc.edu
	wuarchive.wustl.edu

Bulletin Board Systems

Country	*Mailbox*	*Phone number*	*Bps.*
Switzerland	Gepard's Oracle	01 363 70 37	9600
USA	Color Galaxy Milky Way (OS-9/6809 Network, 12 Nodes)	(415) 883 0696	2400
	ACS, Inc. BBS (FIDO 133/510)	(404) 636 2991	???

_getsys(OS-9 International);

Advertisements

OS-9 International is not only an ideal platform for discussing OS-9 related topics, it is also the ideal place to advertise. OS-9 International reaches OS-9 end-users, system-software developers and, nevertheless, decision-makers.

Please contact us for detailed information on how to place an ad in OS-9 International.

Subscriptions

OS-9 International is only available by subscription. We currently offer a subscription of six issues at the following subscription fees:

	CH and FL	*Europe*	*Overseas*
Six issues	CHF 45.00	CHF 68.00	CHF 83.00

To subscribe to OS-9 International send a letter or postcard with your address on it.

We are sure that you understand that subscriptions from outside Switzerland have to be prepaid. This must be done in Swiss currency drawn to our bank account: Account # 10-107,666.0, Marc Balmer, Swiss Bank Corporation, CH-4000 Basel, Switzerland. Your subscription will start upon receiption of your payment.

To subscribers in CH and FL a bill will be mailed together with the next issue.

New E-mail address

The electronic mail address has changed. OS-9 International may be reached at <os9int@msys.ch>.

Code disks

All code presented in this issue is available in electronic form on 3.5" OS-9 universal format disk. This disk may be obtained by sending CHF 20.00 to OS-9 International. Please specify the issue for which you want to receive the code disk.

OS-9 International

ISSN: 1019-6714
Published by Marc Balmer
Editors: Carsten Emde, Marc Balmer